HAMK
Hämeen ammatti-
korkeakoulu

Puheohjattavan kuvageneraattorisovelluksen kehittäminen OpenAI DALL-E 3, React-Speech ja TailwindCSS:llä

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, insinööri (AMK)

Kevät 2025

Ahmet Ari

HAMK
Hämeen ammatti-
korkeakoulu

Koulutuksen nimi Tieto- ja viestintätekniikka — Tiivistelmä
Tekijä Ahmet Ari — Vuosi 2025
Työn nimi Puheohjattavan kuvageneraattorisovelluksen kehittäminen OpenAI DALL-E 3, React-Speech ja TailwindCSS:llä
Ohjaaja Petri Kuittinen

Tässä opinnäytetyössä luotiin sovellus kuvagenerointiin käyttäen äänentunnistusta, ReactJS:ää, TailwindCSS:ää, DALL-E 3 -mallia ja React-Speech-Recognition -kirjastoa. Työn tavoitteena oli kehittää responsiivinen sovellus, joka toimii kaikilla laitteilla ja mahdollistaa kuvien luomisen käyttäjän puheella annettujen komentojen perusteella. Aihe valittiin tekijän oman kiinnostuksen ja puheohjattujen sovellusten kasvavan trendin vuoksi.

Sovellus toteutettiin Reactilla, jossa käyttöliittymän tyylittelyssä hyödynnettiin TailwindCSS:n utility first lähestymistapaa. Äänentunnistus toteutettiin React-Speech-Recognition -kirjaston avulla, ja kuvat generoitiin OpenAI:n DALL-E 3 -mallin avulla API-kutsuin. Sovellusta testattiin paikallisesti Chrome DevTools -työkaluilla.

Lopputuloksena syntyi sovellus, joka täyttää sille asetetut perustoiminnalliset tavoitteet. Sovellus kykenee tunnistamaan käyttäjän puhekomennon, muuntamaan sen tekstiksi ja lähettämään sen DALL-E 3 -mallille, joka palauttaa pyynnön mukaisen kuvan. Kehityksen aikana ilmeni haasteita, kuten samanaikaiset pyynnöt, jotka saatiin ratkaistua hyödyntämällä Reactin useRef-hookia ja peruutusmekanismia.

Johtopäätöksissä todetaan, että alkuperäiset tavoitteet saavutettiin ja tekijä sai arvokasta kokemusta modernien teknologioiden ja tekoälyn yhdistämisestä. Sovelluksen jatkokehitystä ajatellen voitaisiin parantaa selainten välistä yhteensopivuutta, kehittää mobiilisovellusversio sekä lisätä ominaisuuksia, kuten käyttäjätilit ja kuvien tallennusmahdollisuus.

Avainsanat React, TailwindCSS, responsiivisuus, tekoälymalli, prompt
Sivut 32 sivua

HAMK
Hämeen ammatti-
korkeakoulu

DP	Information and Communication Technology	Abstract
Author	Ahmet Ari	Year 2025
Subject	Developing a Speech-Controlled Image Generator Using OpenAI, DALL-E 3, React-Speech, and TailwindCSS	
Supervisor	Petri Kuittinen	

In this thesis, an application was created for image generation using voice recognition, ReactJS, TailwindCSS, DALL-E 3, and the React-Speech-Recognition library. The main goal was to develop a responsive application that works on all devices and generates images through voice commands. The topic was selected based on the author's own interest and the increasing trend of voice-controlled applications.

The application was developed using React, with TailwindCSS used for styling via its utility first approach. Voice recognition was implemented using React-Speech-Recognition, and images were generated using OpenAI's DALL-E 3 model through API requests. The app was tested locally using Chrome DevTools to ensure functionality.

The result is a functional application that meets the basic goals set at the beginning. It successfully recognizes voice commands, converts them into text, and sends the prompt to the AI, which returns the corresponding image. One major issue encountered was overlapping requests, which were solved using React's useRef hook and a cancellation mechanism.

In conclusion, the thesis achieved its original goals and provided the author with valuable experience in integrating modern technologies with AI. For future development, it would be beneficial to improve cross-browser compatibility, build a mobile version using React Native, and add features such as user accounts and image saving functionality.

Keywords React, TailwindCSS, responsiveness, AI-generation model, prompt

Pages 32 pages

Sisällys

Kuvat, taulu kot ja kaavat

1 Johdanto

Puheohjaus ja tekoälypohjaiset kuvageneraattorit, kuten OpenAI:n DALL-E 3, ovat viime vuosina kehittyneet nopeasti ja nousseet osaksi modernia sovelluskehitystä. Nämä teknologiat tekevät sovelluksista käyttäjäystävällisempiä ja tarjoavat uusia mahdollisuuksia esimerkiksi esteettömyyden edistämisessä. Tässä opinnäytetyössä kehitettiin web-sovellus, jossa puheentunnistus yhdistyy tekoälypohjaiseen kuvagenerointiin. Kehitystyössä hyödynnettiin ReactJS-kirjastoa, TailwindCSS-tyylikirjastoa, React-Speech-Recognition-kirjastoa sekä OpenAI:n DALL·E 3 -kuvamallia.

Tavoitteena oli toteuttaa responsiivinen ja yksinkertainen käyttöliittymä, joka toimii eri laitteilla moitteettomasti ja mahdollistaa kuvien luomisen pelkän puhekomennon avulla. Lisäksi työn aikana tarkasteltiin eri teknologioiden yhteensopivuutta ja käytännön toteutusta, erityisesti käyttäjäkokemuksen ja saavutettavuuden näkökulmasta.

Työ pyrkii vastaamaan seuraaviin tutkimuskysymyksiin:

– Miten puheohjaus voidaan tehokkaasti integroida kuvageneraattorisovellukseen Reactin ja OpenAI DALL-E 3:n avulla?

– Miten TailwindCSS vaikuttaa käyttöliittymän kehittämiseen verrattuna perinteiseen CSS:ään?

– Miten eri kuvagenerointimallit eroavat toisistaan kuvanlaadun, tarkkuuden ja tyylin suhteen?

Opinnäytetyö etenee kuvaamalla ensin käytetyt teknologiat ja niiden roolin projektissa. Tämän jälkeen perehdytään tekoälymalleihin ja niiden vertailuun, ja lopuksi esitellään itse toteutus, testaus ja johtopäätökset. Työn tarkoituksena oli tuottaa toimiva, ääntä ymmärtävä kuvageneraattorisovellus sekä tuoda esiin sen kehitysprosessissa syntyneitä oivalluksia ja jatkokehityksen mahdollisuuksia.

2 Teoria

2.1 ReactJS

2.1.1 Yleiskatsaus ReactJS:stä

Facebook julkaisi ReactJS:n ensimmäistä kertaa 29.5.2013, jota on siitä päivästä lähtien jatkuvasti kehitetty, ja on kasvanut sekä noussut huomioon ohjelmoijien parissa. Facebook loi ReactJS:n kasvavan applikaation vuoksi, ja tehokkaampaan tapaan luoda uudelleenkäytettäviä käyttöliittymäkomponentteja, jotka muuttuvat ajan myötä saadun datan mukaan. (Sanket, 2024)

React, yleisemmin kutsuttuna ReactJS:ksi, on JavaScript-kirjasto, joka on tunnettu kyvystään luoda interaktiivisia käyttöliittymiä verkkosivuille, erityisesti single-page-sovelluksiin (SPA), nopeasti ja helposti. Reactin komponenttipohjaisen rakenteen ansioista ohjelmoijat voivat helposti luoda käyttöliittymäelementtejä esim. painikkeita tai hakupalkkeja, jota voidaan käyttää sovelluksessa niin monta kertaa, kun tarvitsee. Tämä ominaisuus tekee kehityksestä johdonmukaista, helpompaa. (Sanity, 2024)

Yksi Reactin keskeisistä ominaisuuksista on JSX (JavaScript XML), joka mahdollistaa HTML-kaltaisen syntaksin käytön JavaScript-koodissa. Tämä mahdollistaa komponenttien järjestämisen ja tekee ohjelmoinnista yksinkertaisempaa (Pete, 2013). Yhdistettynä nämä, komponentit luovat monimutkaisia käyttöliittymiä. Lisäksi jokainen komponentti pystyy käsittelemään omia tietojaan, eli se voi tallentaa ja muuttaa tietoja itseensä. (Sanity, 2024)

Toinen tärkeä ominaisuus on JSX (JavaScript XML), laajennus, jonka avulla kehittäjät voivat kirjoittaa HTML-kaltaista koodia JavaScriptiin. Tämä auttaa kuromaan umpeen JavaScriptin ja HTML:n välistä kuilua ja tekee koodauksesta intuitiivisempaa ja tehokkaampaa. Lisäksi React käyttää virtuaalista DOM:ia (Document Object Model), joka on konsepti, jossa käyttöliittymän ihanteellinen esitys säilytetään muistissa ja synkronoidaan todellisen DOM:n kanssa Reactin täsmäytysprosessin kautta. Tämä tekee verkkosivujen hahmontamisesta erittäin tehokasta, mikä johtaa nopeampiin ja reagoivampiin sovelluksiin. (Sanity, 2024)

Reactin tilanhallinta on joustavaa, ja siinä on sisäänrakennettu tuki luokkakomponenttien "state" tai toiminnallisten komponenttien useState-koukun kautta. React mahdollistaa myös kolmannen osapuolen kirjastot, kuten Redux, suuremmille sovelluksille. (Sanity, 2024)

Sen lisäksi, että React on erinomainen verkkokäyttöliittymien rakentamisessa pääkirjaston react-domilla, se tukee myös mobiilialustan kehitystä react-nativella. Tämä tarkoittaa, että voit luoda sekä verkko- että natiivimobiilisovelluksia käyttämällä samanlaisia koodausmalleja. (Sanity, 2024)

Kun verrataan Reactia muihin suosittuihin JavaScript-kehyksiin, kuten Angular ja Vue.js, yksi asia, joka erottaa sen muista, on sen joustavuus. Toisin kuin Angular ja Vue, joilla on erityisiä tapoja ratkaista ongelmia, React ei pakota ohjelmoijia käyttämään yhtä tiettyä ratkaisua. Tämä antaa ohjelmoijille enemmän vapautta valita, kuinka he haluavat rakentaa ja jäsentää sovelluksiaan. Esimerkiksi Angularilla on oma reititin ja se käyttää NGXS:ää tai NGRX:ää tilanhallintaan, kun taas Vue käyttää Vuexia. Mutta Reactin avulla voit valita useista vaihtoehdoista projektisi tarpeiden mukaan. Reititykseen voit käyttää 'React Routeria' ja tilanhallintaan vaihtoehtoja, kuten Redux tai sisäänrakennettu konteksti-API. Asiakkaan ja palvelimen välisessä viestinnässä sekä Vue.js että React käyttävät usein Axiosta, kun taas Angularilla on oma HTTP Client -moduuli. (Sanity, 2024)

2.1.2 Virtual DOM

Koska verkkosovellukset muuttuvat monimutkaisemmiksi, käyttöliittymän pitäminen ajan tasalla on vaikeaa. Tässä tulee esiin Virtual DOM (Document Object Model) – erityisesti Reactissa, joka on käyttöliittymäsuunnittelun suosituin JavaScript-kirjasto. Virtuaalinen DOM on kevyt kopio todellisesta DOM:sta, joka auttaa Reactia käsittelemään muutoksia tehokkaammin vähentämällä todellisen DOM:n suoraa käsittelyä. Tämä tekee verkkosovellusten toimimisesta paljon sujuvampaa. Virtuaalisen DOM:n tunteminen on erittäin tärkeää kehittäjille, jotka haluavat saada kaiken irti Reactista. Se on iso osa sitä, kuinka React päivittää käyttöliittymää, ja varmistaa, että muutokset tapahtuvat nopeasti ilman tarpeettomia uudelleenrenderöintejä. (Matéu, 2024)

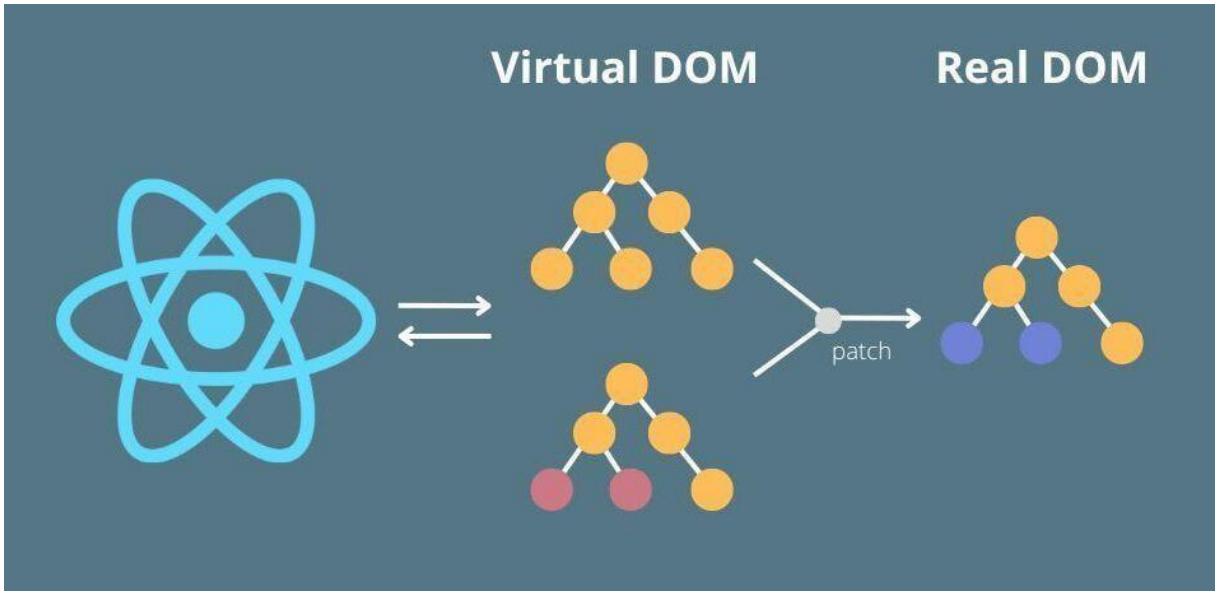
Virtuaalinen DOM on kuin muistiversio todellisista DOM-elementeistä. Sen sijaan, että käsittelisi suoraan todellista DOM:ia, mikä voi olla hidasta ja vaikuttaa suorituskykyyn, React tekee virtuaalisen kopion käyttöliittymäkomponenteista. Tämä virtuaalinen kopio on yksinkertainen JavaScript-objekti, joka näyttää aivan todelliselta DOM:lta. (Fidisys, 2023)

Kuten alhaalla olevasta kuvasta (kuva 1) näemme, keskellä on virtuaalinen DOMi, puinen rakenne kuvaa applikaation nykyhetkeistä tilannetta virtual DOM:ssa. Jokainen solmu symbolisoi React-kompponenttia, tai elementtiä. Tässä kohtaa React seuraa kaikkia tapahtuvat muutokset, kun sovelluksen tila päivittyy, eli rerenderöi. Kaksi puinen rakenne,

jossa ylempi kuvaa aiemman tilan, ja alempi rakenne kuvaa päivitettyä tilaa. Päivitykset ovat kuvan mukaan merkitty punaisella värillä kahden solmun kohdassa.

Virtuaalisen DOMin ja oikean DOMin välillä on kohta patch, joka on harmaalla solmuksella merkitty. Prosessi, jossa React vertailee aikaisemman- ja nykyversion virtual DOMin identifioikseen muutoksen. (Fidisys, 2023)

Kuva 1 Virtual DOM. (Fidisys, 2023)



2.1.3 Komponentit

React komponentteja voidaan luokitella joko funktionaalisiin komponentteihin, tai luokkakomponentteihin. Jokainen palvelee käyttöliittymäelementtien renderöimistä, ja eroaa siitä, että miten ne käsittelevät tila- ja elinkaaritapahtumia. (React, n.d.)

Funktionaaliset komponentit ovat yksinkertaisia JavaScript-funktioita, jotka hyväksyvät rekvisiitta ja palauttavat JSX:n kuvaamaan käyttöliittymää. Aluksi he olivat valtiottomia eivätkä pystyneet hallitsemaan omaa tilaansa tai elinkaartaan, mutta tilanne muuttui React Hooksin käyttöönoton myötä. Koukut, kuten useState ja useEffect, antavat toiminnallisten komponenttien käsitellä tilaa ja sivuvaikutuksia, mikä tekee niistä yhtä tehokkaita kuin luokkakomponentit. Toiminnalliset komponentit ovat yleensä tiiviimpiä, helpompia ymmärtää

ja johtavat usein parempaan suorituskykyyn yksinkertaisuutensa ja tämän kontekstin puuttumisen vuoksi. (GeeksforGeeks & Supraja, 2024)

Näkyvä painike (Kuva 2) on yksinkertainen funktionaalinen komponentti, joka ottaa vastaan ”props”-objektin ja palauttaa JSX-elementin (painikkeen eli buttonin). Props-arvoja hyödyntämällä ReactJS mahdollistaa komponenttien uudelleen käytettävyyden ja luomaan ainutlaatuisia komponentteja.

Kuva 2 ReactJS funktionaalinen komponentti.

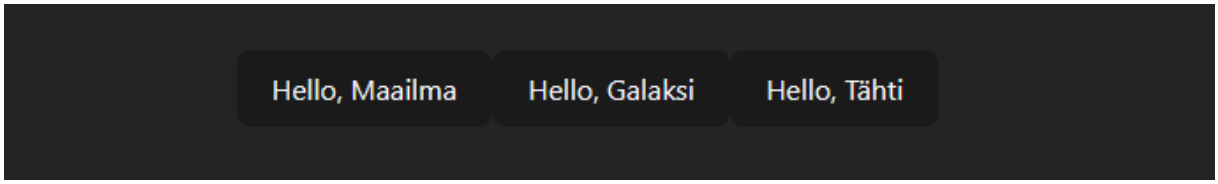
```jsx
import React from "react";

function Painike({ props }) {
  return <button>Hello, {props}</button>;
}

export default Painike;
```

Kuva 3 Komponentin käyttö eri props-arvoilla.

```jsx
import "./App.css";
import Painike from "./komponentit/Painike";

function App() {
  return (
    <>
      <Painike props="Maailma" />
      <Painike props="Galaksi" />
      <Painike props="Tähti" />
    </>
  );
}

export default App;
```

Tuloksena syntyy kolme eri painiketta, joka käyttää hyödyksi props-järjestelmää arvoilla, Maailma, Galaksi, ja tähti.

Kuva 4 Painike-komponentin painike eri props-arvoilla.



Painike-luokan komponentti (Kuva 5) on React-komponentti, joka ottaa tekstipropsin ja tekee sen <h1>-elementin sisällä, ja näyttää "Hei" ja prop-arvon. Komponentti on dynaaminen propsin avulla ja valmiina uudelleenkäyttöön erilaisten viestien näyttämiseen sille välitetyn props-arvon perusteella. Tämä komponentti viedään sitten käytettäväksi muissa React-sovelluksen osissa.

Kuva 5 ReactJS luokkakomponentti.

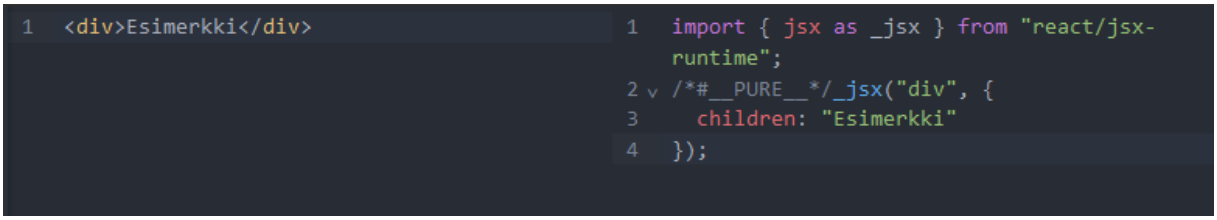
```
import React from "react";

class Painike extends React.Component {
  render() {
    return <h1>Hello, {this.props.teksti}</h1>;
  }
}

export default Painike;
```

2.1.4 Babel

Transpilaattorit ovat kuin kielenmuuntimia ohjelmointia varten, voidaan verrata niitä kielten kääntäjiksi, mutta ohjelmointikielien sisällä. Esimerkiksi Babel muuttaa nykyaikaisen JavaScriptin versioksi, joka toimii kaikilla alustoilla. Babel ja sen kaltaiset transpilaattorit

kuten SWC tai Bublé ovat hyvin samanlaisia toiminnaltaan, mutta käyttötarkoituksen mukaan voi valite itselle sopivin vaihtoehto. Babel on kuitenkin oletuksena CRA (Create React App) työkalussa. Transpilaattorit on tärkeitä, koska kaikki selaimet eivät tue uusinta JavaScriptiä ja kehittäjät tarvitsevat koodinsa toimiakseen kaikkialla. (Sharma, 2024) Babel toimii transpilaattorina ja muuttaa uusimman JavaScriptin vanhemmaksi versioksi, jonka kaikki selaimet ymmärtävät. Näin kehittäjät voivat käyttää uusimpia ominaisuuksia ja varmistaa samalla, että heidän koodinsa on yhteensopiva kaikkien alustojen kanssa. (Aswin, 2019)

Kuva 6 Yksinkertainen JSX transpiloituna Babelissa.



```
<div>Esimerkki</div>
```

```
import { jsx as _jsx } from "react/jsx-runtime";
/*#__PURE__*/_jsx("div", {
  children: "Esimerkki"
});
```

Alemmassa esimerkissä (Kuva 7) nähdään suuremman JSX:n muutoksia. Vasemmalla on alkuperäinen JSX-koodimuoto, ja oikealla on transpiloitu Javascript-versio, joka on luotu Babelilla heidän kotisivussaan.

Kuva 7 Monimutkaisempi JSX:n esimerkki.

```
export default function DiceRoll(){
  const getRandomNumber = () => {
    return Math.ceil(Math.random() * 6);
  };

  const [num, setNum] = useState(getRandomNumber());

  const handleClick = () => {
    const newNum = getRandomNumber();
    setNum(newNum);
  };

  return (
    <div>
      Your dice roll: {num}.
      <button onClick={handleClick}>Click to get a new number</button>
    </div>
  );
};
```

```
import { jsx as _jsx, jsxs as _jsxs } from "react/jsx-runtime";
export default function DiceRoll() {
  const getRandomNumber = () => {
    return Math.ceil(Math.random() * 6);
  };
  const [num, setNum] = useState(getRandomNumber());
  const handleClick = () => {
    const newNum = getRandomNumber();
    setNum(newNum);
  };
  return /*#__PURE__*/_jsxs("div", {
    children: ["Your dice roll: ", num, ".", /*#__PURE__*/_jsx("button", {
      onClick: handleClick,
      children: "Click to get a new number"
    })]
  });
}
;
```

2.2 TailwindCSS

2.2.1 TailwindCSS esittely

TailwindCSS on CSS-kehys, joka tarjoaa laajan valikoiman valmiita tyyliluokkia. Niiden avulla kehittäjät voivat rakentaa käyttöliittymiä suoraan HTML-koodiin ilman erillisten CSS-tiedostojen kirjoittamista. Tailwindin pääfilosofia on tyylitellä komponentteja nopeasti ja

joustavasti käyttämällä valmiita luokkia, kuten bg-blue-500, text-center ja p-4 (Kuva 8). Tailwind eroaa Bootstrapin kaltaisista perinteisistä CSS-kehyksistä siinä, että se ei sisällä valmiita käyttöliittymäkomponentteja (kuten painikkeita tai lomakkeita), vaan antaa sen sijaan kehittäjille täydellisen määräysvallan ulkoasun määrittelyssä. Tailwind välttää monimutkaiset CSS-hierarkiat ja mahdollistaa helposti ylläpidettävän ja modulaarisen koodin. Tailwind tarjoaa myös mobiiliystävälliset responsiiviset ominaisuudet ja mahdollisuuden lisätä mukautettuja tyylejä helposti. (GeeksforGeeks, 2024)

Kuva 8 Yksinkertainen TailwindCSS-tyyli esimerkki.

```
<div>
  <button class="bg-blue-500 text-white px-4 py-2 rounded">
    Klikkaa tästä
  </button>
</div>
```

2.2.2 TailwinCSS vs Perinteinen CSS

CSS on verkkokehityksen peruspilari, jonka avulla määritellään kaikki sivuston tyylit – värityksestä asetteluun. Sen vahvuus on rajaton joustavuus: kehittäjällä on vapaus luoda täysin räätälöityjä designejä ilman ulkoisten rajoitteiden painolastia. Esimerkiksi uniikki animaatio tai monimutkainen grid-järjestelmä onnistuvat CSS:llä, kun osaaja tuntee sen syntaksin ja ominaisuudet. Kuitenkin tämä vapaus vaatii hintansa: suurissa projekteissa omien tyylien kirjoittaminen alusta alkaen on hidasta, ja koodin ylläpito voi muuttua haastelliseksi. Lisäksi responsiivisen designin toteuttaminen mediakyselyillä vaatii usein manuaalista työtä. (Shrihari, 2023)

Yhtenäinen väri- ja tyylikirjasto varmistaa, että design pysyy johdonmukaisena eri näyttökokojen ja komponenttien välillä. Tailwind myös yksinkertaistaa responsiivisuutta: esimerkiksi luokka md:text-xl muuttaa fontin koon automaattisesti tablet-näytöillä. (Shrihari, 2023)

Kuvassa 9 nähdään TailwindCSS:n ja perinteisen CSS:n välisen eron hyvin selvästi, CSS:n käyttö vaatii useita tyylitiedostoja, jotka voivat sekoittaa kehittäjän ajatuksia ja todella pitkät tyylitiedostot vaikeuttavat vain tyylittelyä. TailwindCSS hoitaa saman tyylittelyn yhdellä rivillä samassa tiedostossa, eikä vaadi uutta tiedostoa ja viittä riviä toisin kuin CSS.

Kuva 9 Tailwindillä kirjoitettu koodi, verrattuna CSS-koodiin.

```
<h1 className="text-2xl p-4 font-bold underline esimerkki">Hello world!</h1>
```

```
.esimerkki {
  font-size: 1.5rem;
  line-height: 2rem;
  font-weight: 700;
  text-decoration-line: underline;
}
```

2.2.3 TailwindCSS:n integrointi React-projekteihin

TailwindCSS:n käyttö React-projekteissa on hyvin yksinkertainen, ja sen sisältämät vaiheet ovat hyvin selitettyjä heidän kotisivuillaan. Annetaan esimerkin asentamalla Tailwindiä heidän Framework oppaalla Vite:n kanssa. Ensimmäiseksi syötetään ”npm create vite@latest my-project -- --template react” komento terminaalille, joka luo meille hakemiston my-project nimellä. Siirrytään meidän hakemistoon kirjoittamalla “cd my-project”. Tärkein komento on tämä seuraava vaihe “npm install -D tailwindcss postcss autoprefixer”, jossa asennetaan Tailwindiä, postcss:ää ja autoprefixeriä. Tämän jälkeen asennuksen viimeistely tapahtuu CLI:n avulla. Komento on “npx tailwindcss init -p”, joka antaa meille “tailwind.config.js” tiedoston (Kuva 10), jota pitää muokata ohjeen mukaisesti. (TailwindCSS, n.d.)

Kuva 10 Ohjeen mukaan kopioidun tailwind.config.js tiedoston lopputulos.

tailwind.config.js

my-project > tailwind.config.js > ...

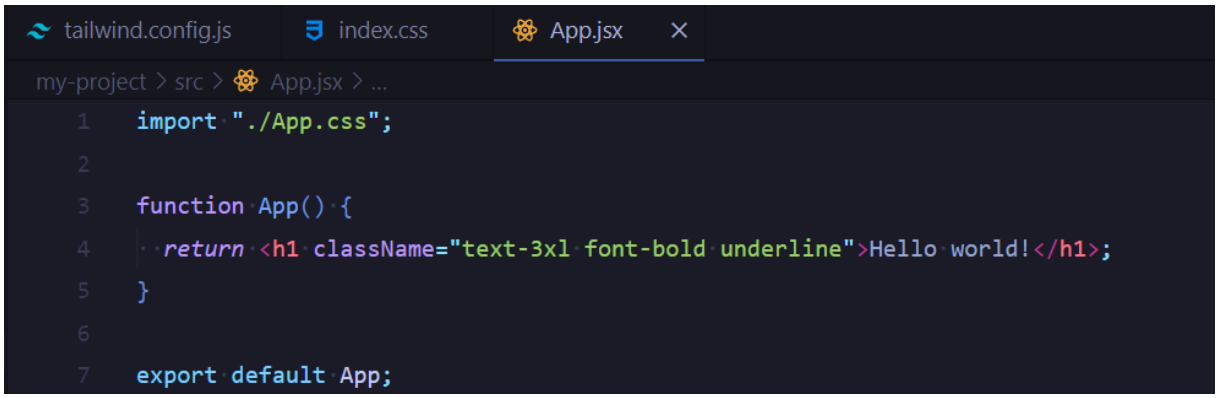
```
/** @type {import('tailwindcss').Config} */
export default {
  content: ["./index.html", "./src/**/*.{js,ts,jsx,tsx}"],
  theme: {
    extend: {},
  },
  plugins: [],
};

```

Vite asennuksen yhteydessä tulevaa index.css tiedostoa src hakemistossa, src\index.css (Kuva 11). Poistetaan kaikki aikaisemmat koodit sieltä, ja kirjoitetaan ohjeen mukaista kolmen rivin koodia. (TailwindCSS, n.d.)

Kuva 11 Muokattu index.css.

Kuvassa 12 on h1 elementti, jonka className:lle on kirjoitettu Tailwindcss:llä eri tyylejä, ensimmäinen tyyli muuttaa fontti koon ja rivivälin, toinen lihavoi tekstin, ja viimeinen tyyli alleviivaa. Kun kaikki vaiheet on onnistuneesti suoritettu, voidaan käynnistää sovellus komennolla “npm run dev” ja nähdä muutokset.

Kuva 12 App.jsx tiedoston lopputulos.



2.2.4 Tyylien hallinta ja mukauttaminen

TailwindCSS on tyylikirjasto, joka toimii utility-first-tyylillä. Tämän ansiosta kehittäjät voivat käyttää valmiita pieniä tyyliluokkia, kun he rakentavat käyttöliittymiä nopeasti. Tämä on erilaista verrattuna perinteiseen CSS:ään, jossa tyylit pitäisi määritellä erikseen ja liittää sitten elementteihin luokkien avulla. TailwindCSS tekee tyylien hallinnasta tosi helppoa ja antaa mahdollisuuden muokata projektin ulkoasua ilman vaivannäköä. (Anna, 2022)

Yksi parhaimmista ominaisuuksista Tailwindissä on sen mukautettavuus. Kehittäjät voivat helposti säätää Tailwindin perusmuotoilua omaan makuunsa lisäämällä omia asetuksia tailwind.config.js-tiedostoon. Esimerkiksi jos sun yrityksellä on omat brändivärit, ne voi kätevästi lisätä Tailwindin omaan värivalikoimaan (Anna, 2022). Näin koko projekti saa kivan ja yhtenäisen ilmeen, ja samat värit on helppo napata käyttöön missä vaan koodissa. Mukautettavuutta löytyy myös muista jutuista, kuten fonteista, varjoista ja marginaaleista. Tämä tekee Tailwindista todella monipuolisen työkalun kaikenlaisiin projekteihin.

TailwindCSS tekee responsiivisten tyylien luomisesta tosi helppoa. Nykyään verkkokehityksessä on tosi tärkeää, että sivustot toimivat hyvin monilla eri laitteilla, kuten puhelimilla, tableteilla ja tietokoneilla. Tailwindissa on käteviä valmiita luokkia, kuten sm:, md: ja lg:, joilla voit säätää tyylejä eri kokoisille näytöille. Esimerkiksi voit muuttaa tekstin kokoa niin, että se on pienillä ruuduilla helposti luettavaa, mutta suuremmilla se voi olla näyttävämpää. Näin käyttöliittymä mukautuu saumattomasti eri laitteisiin. (TailwindCSS, n.d.)

Kuva 13 JSX-koodin pätkässä hyödynnetään TailwindCSS:n luokkia h1-elementin tyylitykseen. Jokainen className-ominaisuuden sisällä oleva luokka määrittelee jonkin visuaalisen tyylin, mikä tekee komponentin ulkoasusta helposti hallittavan ja selkeän.

- ***text-3xl***, tämä määrittää tekstin kooksi 3xl, joka on TailwindCSS:n esiasetettu fonttikoko. Se tekee tekstistä suuren ja näkyvän. Tailwindin fonttikoot seuraavat johdonmukaista asteikkoa.

- ***font-bold***, tekee tekstistä lihavoitua (bold). Se korostaa h1-otsikon merkitystä ja tekee siitä huomattavan.

- ***Underline***, lisää alleviivauksen tekstiin. Tämä tyyli on hyödyllinen esimerkiksi otsikoissa tai linkeissä, joissa alleviivaus kiinnittää käyttäjän huomion.

- text-cyan-500, määrittää tekstin väriksi syaanin (turkoosin sävyn). Numero 500 viittaa TailwindCSS:n väripalettiin, jossa värit ovat jaoteltu asteikon mukaan vaaleasta tummempaan (esim. 100 = vaalea, 900 = tumma). Tässä tapauksessa 500 tarkoittaa keskisävyä.

Kuva 13 H1-elementin mukauttaminen TailwindCSS:llä.

```
<h1 className="text-3xl font-bold underline text-cyan-500">Hello world!</h1>
```

2.3 OpenAI API ja React Speech-to-text

2.3.1 Yleistietoa OpenAI:sta

OpenAI on tekoälytutkimuslaboratorio, joka perustettiin vuonna 2015. Sen tavoitteena oli kehittää tekoälyä ja varmistaa, että se hyödyttää kaikkia. OpenAI:lla on monta perustajaa mutta, muutama tunnetuimmista toisista perustajista ovat Elon Musk ja Sam Altman. (Coursera, 2024)

OpenAI:lla on Monia erilaisia palveluita, jotka helpottavat arikipäivää ihmisille, ohjelmoijille, jopa taiteilijoita. Nämä palvelut ovat: ChatGPT (chattibotti vastaa prompteihin tai kysymyksiin) Dalle-E 3(Kuvagenerointimalli), Codex (Niinkuin ChatGPT mutta ohjelmoijille tarkoitettu tekoäly), Whisper(Puheentunnistus tekoäly joka pystyy myös kääntämään kieliä), Scholar (Ohjelma joka tukee tutkijoita, opiskelijoita). Seuraavassa osioissa käydään tarkemmin läpi useita palveluita. (Coursera, 2024)

2.3.2 OpenAI API

Tekoälyteknologia on mullistanut digitaalisen maailman ongelmien ratkaisun ja uusian ongelmien luomisen osalta. OpenAI API on yksi parhaista työkaluista, jolla GPT:n, Codexin, ja Whisperin kaltaiset kehittyneet tekoälyteknologiat saadaan helposti ohjelmistokehittäjien, ja yritysten käyttöön (Coursera, 2024). OpenAI API:n avulla kehittäjät voivat rakentaa monimutkaisempia tekoälysovelluksia ilman, että heidän tarvitsee kouluttaa omia mallejaan. Tämä helpottaa tekoälyratkaisujen käyttöä ja nopeuttaa uusien ideoiden luomista.

OpenAI API on helppo integroida sovelluksiin ohjelmointirajapinnan kautta. Esimerkiksi JavaScriptissä API:n käyttö onnistuu käyttämällä fetch-metodia tai Axios-kirjastoa.

Kuvassa 14 on yksinkertainen esimerkki siitä, miten API:ta voidaan käyttää tekstin generointiin. Ensin luodaan yhteys OpenAI API:lle fetch-funktiolla, sitten määritetään parametrit joka sisältää myös API-avaimen, asetetaan parametrit(malli, käyttäjän syöte, tuotetun tekstin kustannuksen hallinta), vastaanotetaan vastaus JSON-muodossa, haetaan teksti vastaus ja tulostetaan tulos.

Kuva 14 OpenAI API:n käyttö fetch-metodilla.

```
const fetch = require("node-fetch");

const fetchResponse = async (prompt) => {
  const response = await fetch("https://api.openai.com/v1/completions", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer YOUR_API_KEY`,
    },
    body: JSON.stringify({
      model: "gpt-4",
      prompt: prompt,
      max_tokens: 100,
    }),
  });
  const data = await response.json();
  return data.choices[0].text.trim();
};

// Käyttö:
fetchResponse("Miten OpenAI API toimii?")
  .then((response) => console.log(response))
  .catch((error) => console.error(error));
```

Tulevaisuudessa OpenAI API:n odotetaan kehittyvän entistä tehokkaammaksi ja saavutettavammaksi. GPT-4 tuli vuonna 2023 josta kehittyi tällä hetkellä viimeisin flagship malli GPT-4o. (OpenAI, n.d.)

2.3.3 React-speech-recognition puheentunnistukseen

Reach-speech-recognition toimii käyttämällä Web Speech API:a kaapatakseen ääntä käyttäjän mikrofonista ja muuntaa sitä kaapattua audiota tekstiksi. Kyseinen pakettii on julkaistu npm sivustolla 2 vuotta sitten. Npm on maailman suurin ohjelmistorekisteri, jonka github osti vuonna 2020 (Nat, 2024). Kehittäjät voivat kätevästi hallita tämän API:n tilaa ja puheentunnistuksen juttuja, kuten sen käynnistämistä, pysäyttämistä ja nollaamista, käyttämällä useSpeechRecognition-hookia. Nämä toiminnot on helppo sisällyttää React-

komponentteihin, mikä tekee ääniohjattujen käyttöliittymien rakentamisesta tosi vaivatonta. Kirjaston asentaminen npm:n kautta ja sen komponenttien tuominen tekee puheentunnistuksen lisäämisestä sekä pieniin että suuriin projekteihin on superhelppoa. (James, 2022)

Kuvassa 15 on yksinkertainen esimerkki komponentista, joka käyttää tätä koukkua. Ensiksi tuodaan SpeechRecognition ja useSpeechRecognition kirjastosta, määritämme Dictaphone-funktion, joka mahdollistaa käyttäjäen puheen tunnistuksen ja sen esittämisen tekstinä. Seuraavaksi teemme selaintuen tarkistuksen, jos kyseinen selain ei tue niin ilmoitamme siitä käyttäjälle. Viimeisessä kohdassa on koodi käyttöliittymää varten teksti, joka muuttuu tilan mukaan, aloita(start), lopeta(stop), nollaa(reset) näppäimet, ja vihdoinkin transkriptio. (Soham, 2024)

Kuva 15 ReactJS sovellus puheentunnistukseen.

```
import React from 'react';
import SpeechRecognition, { useSpeechRecognition } from 'react-speech-recognition';

const Dictaphone = () => {
  const {
    transcript,
    listening,
    resetTranscript,
    browserSupportsSpeechRecognition
  } = useSpeechRecognition();

  if (!browserSupportsSpeechRecognition) {
    return <span>Browser doesn't support speech recognition.</span>;
  }

  return (
    <div>
      <p>Microphone: {listening ? 'on' : 'off'}</p>
      <button onClick={SpeechRecognition.startListening}>Start</button>
      <button onClick={SpeechRecognition.stopListening}>Stop</button>
      <button onClick={resetTranscript}>Reset</button>
      <p>{transcript}</p>
    </div>
  );
};
```

2.4 AI-Kuvagenerointimallit

2.4.1 Mitä ovat AI-Kuvagenerointimallit

Tekoäly (AI) on kehittynyt merkittävästi viime vuosina, erityisesti kuvien tuottamisen alalla. Tekoälyn kuvantuotantomalleista ja -työkaluista on tullut yhä kehittyneempiä, ja niiden avulla käyttäjät voivat luoda korkealaatuisia kuvia tekstikuvauksista. Näillä edistysaskelilla on laajoja sovelluksia eri toimialoilla, kuten graafisessa suunnittelussa, videopelien kehittämisessä, markkinoinnissa ja mainonnassa.

OpenAI:n DALL-E on näkyvä esimerkki tällaisesta teknologiasta. Se on kerännyt huomattavan käyttäjäkunnan, ja yli 1,5 miljoonaa käyttäjää tuottaa päivittäin yli kaksi miljoonaa kuvaa. DALL-E toimii freemium-mallilla, jossa käyttäjille tarjotaan aluksi ilmaisia krediittejä ja mahdollisuus ostaa lisää krediittejä. (Ben, 2023)

Huolimatta nopeasta kehityksestä ja näiden työkalujen hyödyllisyydestä on tärkeää tiedostaa mahdolliset haasteet. Tekoälyn tuottamassa sisällössä voi joskus esiintyä harjoitteluaineistosta johtuvia vääristymiä tai epätarkkuuksia. Siksi tekoälyn tuottama sisältö olisi tarkistettava perusteellisesti ihmisen toimesta tarkkuuden ja asianmukaisuuden varmistamiseksi.

2.4.2 Suosituimmat AI-Kuvageneraattorit

Tekoälykuvageneraattorit ovat aika siistejä, koska ne käyttävät koulutettuja keinotekoisia neuroverkkoja luodakseen kuvia tyhjästä. Ne voivat luoda alkuperäisiä ja realistisia kuvia käyttäjän antamien tekstipohjaisten ohjeiden perusteella. Erityisen houkuttelevaa on se, että ne voivat yhdistellä erilaisia tyylejä, käsitteitä ja ominaisuuksia, jolloin syntyy taiteellisia ja asiayhteyteen sopivia kuvia. Tämä kaikki onnistuu Generatiivisen tekoälyn avulla, joka on nimenomaan suunniteltu sisällön luomiseen.

Tekoälyn kuvageneraattorit koulutetaan laajalla datamassalla, joka sisältää paljon erilaisia kuvia. Kun algoritmeja koulutetaan, ne oppivat tunnistamaan kuvien eri juttuja ja piirteitä. Niinpä ne pystyvät luomaan uusia kuvia, jotka näyttävät ja tuntuvat samankaltaisilta kuin ne, mitä treenisetissä on. (Adam, 2023)

On olemassa kaikenlaisia tekoälykuvageneraattoreita, ja jokaisella niistä on omat erikoiset juttunsa. Esimerkiksi on tämä neuraalinen tyylinsiirtotekniikka, joka siirtää toisen kuvan tyylin toiseen kuvaan, sitten on nämä generatiiviset vastakkaisverkot (eli GANit), joissa on

kaksiosainen neuroverkko, joka treenaa luomaan tosi aidon näköisiä kuvia, jotka näyttävät samalta kuin mitä sillä on koulutuksessa käytetty. Sitten on diffuusiomallit, jotka luovat kuvia prosessilla, joka jäljittelee hiukkasten liikettä ja muuttaa kohinan vähitellen selkeäksi kuvaksi. Vertaillaan näitä malleja seuraavassa kappaleessa. (Sherlock, 2025)

2.4.3 Vertailu eri tekoälymallien kesken

Tekoälymalleja on monta, tämän opinnäytetyön tarkoituksena on kuitenkin kuvien luomista tekoäly-kuvageneraattoreille, joten keskitytään näihin. Tekoälypohjaiset-kuvageneraattorit käyttävät pääasiassa kahta malliperhettä, jotka ovat: Diffuusiomalleja ja GAN-malleja, mutta muitakin tekniikoita käytetään mukana.

Diffuusiomallit ovat viime aikoina nousseet erityisesti suosioon kuvageneroinnissa, verrattuna GAN-malliin on hitaampi luomaan kuvia. Kuvageneraattorit jotka käyttävät diffuusiomalleja ovat: Stable Diffusion (Stability AI), DALLE 2 & 3, MidJourney, Imagen (Google DeepMind).

Generative Adversarial Network (GAN) toimii kouluttamalla kahta verkkoa samanaikaisesti: generaattoria ja erottajaa. Generaattori on vastuussa synteettisten kuvien tuottamisesta, kun taas erottelija toimii luokitusverkostona, joka arvioi kuvat sen määrittämiseksi, ovatko ne oikeita vai vääriä. Molemmat verkostot käyvät koulutusta samanaikaisesti; Diskriminaattori pyrkii luokittelemaan tarkasti aidot ja keksityt kuvat, kun taas generaattori pyrkii huijaamaan erottelijan luokittelemaan väärennetyt kuvat aitoiksi väärin. (Adriel, n.d.)

Kuvassa 16 Nähdään koulutusepookkien määrä ja miten paljon se vaikuttaa kuvien laatuun, jokainen määrä kertoo kuinka monta kertaa malli käy läpi tietokokonaisuuden.

Kuva 16 Koulutusepookkien määrä ja sen vaikutus kuvien laatuun (Adriel, n.d.).

Images generated by generator over time.

GANin edut:

- Tuottaa korkealaatuista, realistisia kuvia, videoia ja äänitallenteita
- Parantaa kuvia ja sisällön luomiseen

GANin miinukset:

- Tehokas kouluttaminen edellyttää merkittäviä laskentaresursseja
- Tuotettu sisältö herättää ettisiä pulmia erityisesti väärennösten luomisessa

Diffuusiomallin kouluttaminen aivan kuten GANin kouluttamiseen tarvitaan suuria määriä dataa, kuten esimerkiksi. Kuvia, ääniä, diffuusiomallit pystyvät matkimaan koulutetun datan pohjalta. (AltexSoft, 2023)

Diffuusiomallien edut:

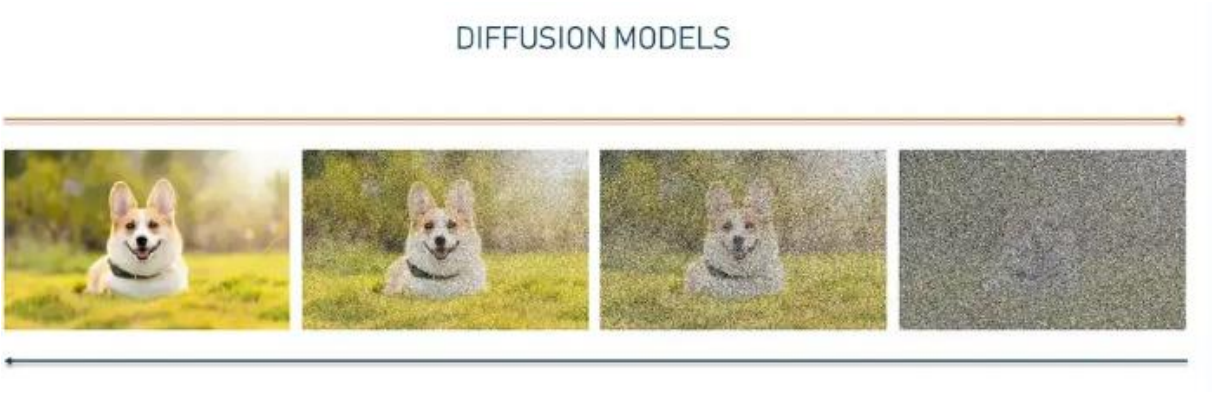
- Korkealaatuisia tuloksia, jotka useimmiten ovat parempia verrattuna GANiin realismin kannalta
- Luotettavampi koulutus prosessi verrattuna GANiin

Diffuusiomallien miinukset:

- Vaatii valtavasti resursseja koulutukseen
- Datan tulosten luominen vie enemmän aikaa verrattuna GANiin

Vähiten kohinan lisääminen, kunnes kuva muuttuu täysin kohinaksi (Kuva 17). Verkko on koulutettu sitten käänteisprosessilla, jossa se arvioi kuvan, jossa on vähemmän kohinaa verrattuna edelliseen kuvaan. (Adriel, n.d.)

Kuva 17 Kohinan lisääminen kuvalle (AltexSoft, 2023).



Kuten kuvasta 18 havaitsemme, vasemmalla puolella on diffuusiomallin luoma kuva ja oikealla puolella GAN-mallin luoma kuva. Promptina käytettiin” photo of a european medieval 40-year-old queen, silver hair, highly detailed face, detailed eyes, head shot, intricate crown, age spots, wrinkles”. Kuvista huomataan, että Diffuusiomallin luoma tulos on realistisempi verrattuna toisen mallin luomaan kuvaan.

Kuva 18 Diffuusiomallin ja GANin välinen ero.

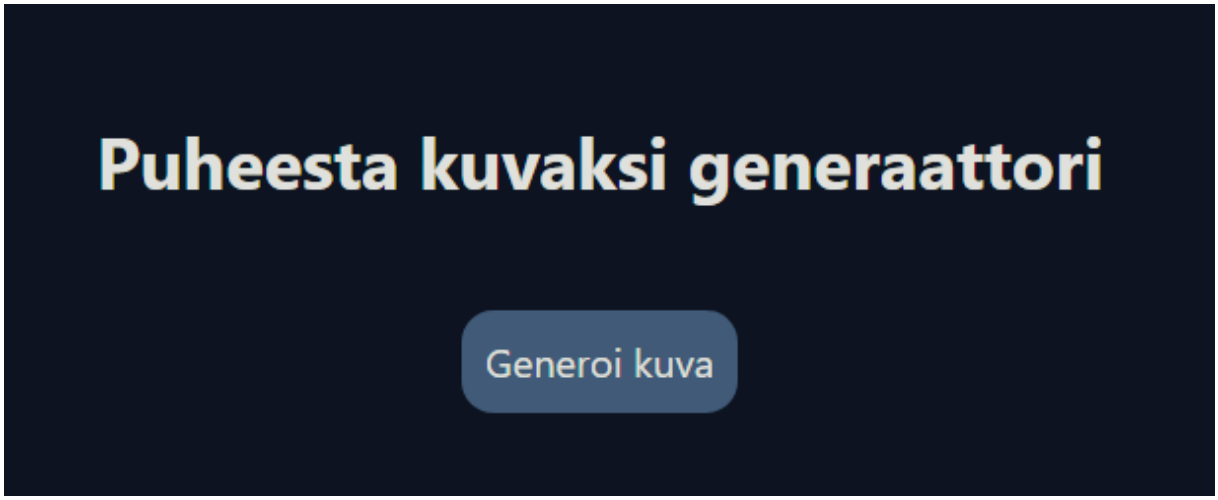
3 Käytännön toteutus

3.1 Projektin rakenne

Tässä opinnäytetyössä rakennetaan moderni, puheohjattava kuvageneraattorisovellus, joka hyödyntää tekoälyä ja web-teknologioita. Idea on yksinkertainen: käyttäjä sanoo ääneen, mitä haluaa nähdä, ja sovellus muuntaa puheen tekstiksi, joka syötetään OpenAI:n DALL-E 3 -mallille. Lopputuloksena saadaan tekoälyn luoma kuva käyttäjän toiveiden pohjalta.

Sovelluksen käyttöliittymä on suunniteltu minimalistiseksi ja helppokäyttöiseksi. Käyttäjän huomio ohjataan suoraan päätoimintoon – puheen avulla kuvien luomiseen. Näytöllä on tumma tausta, vaalea otsikko "Puheesta kuvaksi generaattori" ja yksi selkeä toimintopainike: "Generoi kuva" (Kuva 19). Kun painiketta painetaan, sovellus alkaa kuunnella käyttäjän puhetta ja muuntaa sen tekstiksi, jonka jälkeen kuva luodaan tekoälyn avulla.

Kuva 19 Projektin käyttöliittymä.



Sovelluksen rakentamisessa käytetään ReactJS:ää, joka on suosittu ja tehokas JavaScript-kirjasto käyttöliittymien kehittämiseen. React mahdollistaa komponenttipohjaisen arkkitehtuurin, eli sovelluksen eri osat voidaan kehittää ja muokata helposti ilman, että koko järjestelmää tarvitsee uudistaa. Käyttöliittymän ulkoasun tyylittelyyn käytetään TailwindCSS:ää, joka tekee CSS:n kirjoittamisesta nopeaa ja joustavaa valmiiden tyyliluokkien avulla. Tailwindillä voi esimerkiksi muuttaa värejä, fontteja ja painikkeiden ulkoasua suoraan HTML-rakenteessa ilman erillisiä CSS-tiedostoja.

Sovelluksen JSX-koodi on jaettu kolmeen osaan -elementeillä (Kuva 20), ensimmäinen osio on sovelluksen otsikko H1-elementti ”Puheesta kuvaksi generaattori”, jota on tyylitelty

TailwindCSS:llä, toiseksi on generoidun kuvan näyttäminen, aluksi mitään ei ole näkyvissä koska mitään kuvaa ei ole generoitu, vasta kun lähetetään prompti tekoälylle ja saadaan takaisin vastauksen niin kuva tulee näkyviin, kolmanneksi on painike, jolle painamalla sovellus kuuntelee käyttäjän mikrofonin ja puhumisen päädyttyä lähettää selville saadun puheen tekoälylle promptina generoitavaksi. Esimerkiksi jos käyttäjä sanoo ”Lihava poliisi syömässä donitseja”, niin tekoäly generoi sen mukaisen kuvan.

Kuva 20 Projektin JSX-rakenne.

```jsx
return (
  <div className="bg-[#0D1321] h-screen mx-auto flex flex-col items-center justify-center">
    <div className="py-4 my-4">
      <h1 className="text-[#E0E1DD] text-center text-4xl font-bold">
        Puheesta kuvaksi generaattori
      </h1>
    </div>
    <div className="">
      {imageUrl && (
        <img
          src={imageUrl}
          alt="AI generoitu kuva"
          width="700"
          height="700"
          className="px-4"
        />
      )}
    </div>
    <div className="my-6 text-center">
      <button
        className="bg-[#415A77] text-[#E0E1DD] text-xl rounded-2xl p-3"
        onClick={listening ? stopListening : startListening}
      >
        {listening ? "Lopeta kuunteleminen" : "Generoi kuva"}
      </button>
      {listening && (
        <p className="text-white text-sm">Kuunellaan... Puhu nyt.</p>
      )}
      {transcript && (
        <p className="text-white text-sm py-2">Teksti: {transcript}</p>
      )}
      {loading && <p className="text-white text-sm">Generoidaan kuvaa...</p>}
    </div>
  </div>
```

3.2 Projektin toiminta

Puheentunnistuksen käyttö helpottaa sekä tekoälysovellusten että muiden sovellusten käyttöä, erityisesti lapsille ja vanhuksille. Helppokäyttöisyyden huomioon ottaminen ja minimalistinen sovellus olivat tärkeitä.

3.3 Testaus ja virheenkorjaus

Sovelluksen koodin kirjoittamisen aikana luulin jo olevani valmis, joten aloin etsiä bugeja, jotka saattaisivat rikkoa ohjelman. Kovan työskentelyn jälkeen havaitsin yhden suuren ongelman, joka on, kun käyttäjä on lähettänyt promptin tekoälylle. Jos tekoälyllä on jo yksi generoitava kuva jonka se työstää ja käyttäjä ehtii lähettämän heti toisen perään, niin se generoi niitä eri aikaan, jolloin viimeisenä generoitu kuva ilmenee ensimmäisenä generoidun kuvan tilalle jolloin se katoaa. Tämä tilanne piti korjata niin, että jos käyttäjä antaa toisen promptin heti perään, sovelluksen täytyy havaita se, peruuttaa ensimmäinen pyyntö ja jatkaa uuden pyynnön käsittelyä.

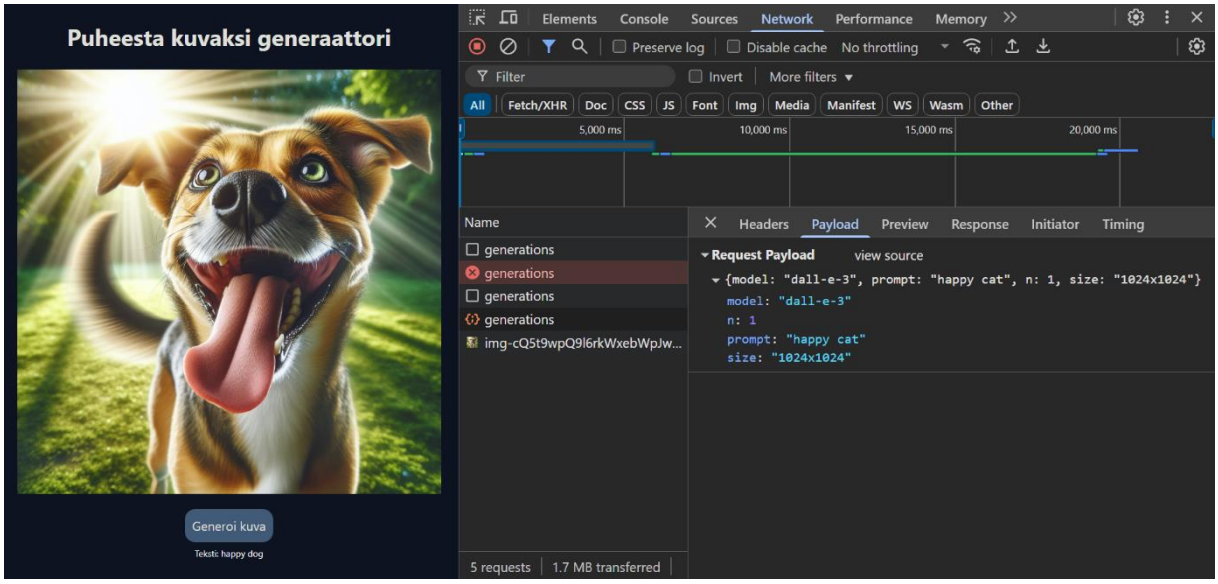
Kuvassa 21 Reactin useRef-hookki on juuri sopiva työkalu tämän ongelman korjaamiseksi, eikä vaadi sovelluksen uudelleenrenderöintiä sen arvon muuttuessa. Kun käyttäjä aloittaa puheentunnistuksen, luodaan cancelTokenRef ja alustetaan se null arvolla. CancelTokenRef.current tokeni mahdollistaa pyynnön peruttamista.

Kuva 21 Pyyntöjen hallinta.

```
const stopListening = async () => {
  if (cancelTokenRef.current) {
    cancelTokenRef.current.cancel("Operation cancelled by user");
    cancelTokenRef.current = null;
  }

const response = await axios.post(
  "https://api.openai.com/v1/images/generations",
  {
    model: "dall-e-3",
    prompt: transcript,
    n: 1,
    size: "1024x1024",
  },
  {
    headers: {
      Authorization: `bearer sk-proj-H9n3em3f6AuGqmA0yBmExshcSGO
    },
    cancelToken: cancelToken.token,
  }
```

Kuvassa 22 nähdään että aiempi pyyntö ”Happy cat” on peruutettu ja sen tilalle uudempi pyyntö ”Happy dog” on generoitu näytöllä.

Kuva 22 Chrome-työkalujen avulla havaittu peruutuspyyntö.



4 Johtopäätökset

Opinnäytetyön alkuperäiset tavoitteet oli kehittää puheohjattu kuvageneraattorisovellus, joka käyttää tekoälyä tässä tapauksessa Open AI:n DALL-E 3 mallia. Sovellus toteutettiin seuraavilla teknologioilla: ReacJS:llä, TailwindCSS:llä, ja React-speech-recognition kirjastoa äänentunnistukseen.

Työn päätavoitteet olivat seuraavat:

- Luoda toimiva ja responsiivinen käyttöliittymä.
    - Tämä onnistui hyvin, sovellus saatiin toimimaan moitteettomasti eri näyttöresoluutoilla, esim. Puhelimeilla, tableteitta.
- Sovellus, joka generoi kuvan äänikomennoilla.
    - Kuvagenerointi onnistui sovelluksella, alussa havaittiin ongelmia, joita ratkaisiin ja saatiin toimiva kokonaisuus.
- Kuvagenerointimallien vertailu.
    - Työssä haluttiin myös vertailla eri AI-kuvagenerointimalleja niiden kuvanlaadun, tarkkuuden ja tyylin perusteella.

Työtä kehittäessäni opin paljon erilaisia tekniikoita ja niiden yhdistämisestä toimivan sovelluksen luomiseksi. TailwindCSS:n käyttö tässä työssä verrattuna perinteiseen CSS:ään

helpotti työtäni merkittävästi. Pääsin samalla syventymään TailwindCSS:ään ja huomasin pitäväni siitä todella paljon. Sen lisäksi API:n hallinnasta opin todella paljon, pyynnön lähettämisestä sekä sen peruuttaminen on yksi tärkeimmistä taidoista minkä opin. Sovelluksen testaaminen eri laitteilla ja selaimen kehitystyökaluilla oli myös erittäin hyödyllistä. Niiden avulla löydettiin yhden merkittävän virheen, jonka pystyi korjaamaan heti.

Tällä hetkellä kyseinen sovellus toimii vain useammalla selaimella, koska kaikki selaimet eivät tue SpeechRecoginition API:a, parhaan kokemuksen saa käyttämällä Chromea.

Jatkokehitysideat:

1. Selainyhteensopivuuden parantaminen
    - Polyfillin käyttö, jotta sovellus toimisi paremmin selaimilla, jotka eivät tue SpeechRecognitionia.
2. Mobiilisovelluksen kehittäminen
    - React Native mahdollistaisi sovelluksen käytön suoraan puhelimella, valmis sovellus voi julkaista play kauppaan.
3. Käyttäjätilit ja tallennusmahdollisuus
    - Käyttäjät voisivat rekisteröityöä sovellukseen, ja tallentaa luomansa kuvat tai halutessa mahdollistaa niitten myöhemmän katselun.

Lopuksi mielestäni opinnäytetyö onnistui hyvin saavuttamaan alkuperäiset tavoitteet ja tarjosi minulle arvokasta oppimiskokemusta modernien teknologioiden ja tekoälyn yhdistämisestä. Valitut teknologiat – ReactJS, TailwindCSS, OpenAI API ja React-Speech-Recognition tukivat sovelluksen toimivuutta ja käytettävyyttä. Työ tarjosi arvokasta oppimiskokemusta API-hallinnasta, käyttöliittymäsuunnittelusta ja tekoälymallien hyödyntämisestä.

Lähteet

1. Adriel, L. (n.d.). GANs vs. Diffusion Models: Putting AI to the test. Haettu 20.02.2025 osoitteesta https://tinyurl.com/5f7m4hrj

2. Adam, Z. (9.11.2023). Explained: Generative AI. Haettu 27.01.2025 osoitteesta https://news.mit.edu/2023/explained-generative-ai-1109

3. AltexSoft. (10.72023). AI image generation Explained: Techniques, Applications, and Limitations. Haettu 20.02.2025 osoitteesta https://www.altexsoft.com/blog/ai-image-generation/

4. Anna, F. (25.5.2022). Tailwind CSS: What It Is, Why Use It & Examples. Haettu 15.10.2024 osoitteesta https://blog.hubspot.com/website/what-is-tailwind-css

5. Aswin, G. (11.11.2019). Webpack and Babel: What are they and how to use them with React. Haettu 3.09.2024 osoitteesta https://tinyurl.com/3yx6fzt7

6. Ben, W. (26.72023). AI Image-Generation Models and Tools: The Ultimate List. Haettu 25.01.2025 osoitteesta https://tinyurl.com/mr224nyh

7. Coursera, S. (2024). What is OpenAI? Everything you need to know. Haettu 20.11.2024 osoitteesta https://www.coursera.org/articles/what-is-openai

8. Fidisys (14.9.2023). React virtual DOM. LinkedIn. Haettu 24.08.2024 osoitteesta https://www.linkedin.com/pulse/react-virtual-dom-fidisys

9. GeeksforGeeks. (2024). Introduction to Tailwind CSS. Haettu 10.10.2024 osoitteesta https://www.geeksforgeeks.org/introduction-to-tailwind-css/

10. GeeksforGeeks. (2024). *ReactJS Functional Components*. Haettu 25.08.2024 osoitteesta https://www.geeksforgeeks.org/reactjs-functional-components/

11. James, B. (10.8.2022). react-speech-recognition. Haettu 3.12.2024 osoitteesta https://www.npmjs.com/package/react-speech-recognition#supported-browsers

12. Matéu, S. (5.6.2024). What is the virtual DOM in React? Haettu 24.08.2024 osoitteesta https://www.freecodecamp.org/news/what-is-the-virtual-dom-in-react/

13. Nat, F. (16.3.2024). npm is joining GitHub. Haettu 3.12.2024 osoitteesta https://tinyurl.com/37fkt6tx

14. OpenAI. (n.d.). Model documentation. Haettu 20.11.2024 osoitteesta https://platform.openai.com/docs/models/o1

15. Pete, H. (5.62013). Why did we build React? React Blog. Haettu 23.08.2024 osoitteesta https://legacy.reactjs.org/blog/2013/06/05/why-react.html

16. React. (n.d.). Components and props. React Documentation. Haettu 25.08.2024 osoitteesta https://legacy.reactjs.org/docs/components-and-props.html

17. Sanket, S. (2024). Decoding the evolution of React: Facebook innovation journey. Haettu 23.08.2024 osoitteesta https://tinyurl.com/yzu7k5fw

18. Sanity. (23.8.2024). *React.js*. Sanity Glossary. Haettu 24.08.2024 osoitteesta https://www.sanity.io/glossary/react-js

19. Sharma, D. R. (10.1.2024). Decoding JavaScript Transpilers: A Babel and Its Alternatives. Haettu 3.09.2024 osoitteesta https://codewithsharma.medium.com/babel-the-javascript-compiler-57f081e9124a

20. Sherlock, X. (23.1.2025). A guide to open-source image generation models. Haettu 25.01.2025 osoitteesta https://www.bentoml.com/blog/a-guide-to-open-source-image-generation-models

21. Shrihari, M. (21.3.2023). CSS vs Tailwind CSS. Haettu 15.10.2024 osoitteesta https://medium.com/@shriharim006/css-vs-tailwind-css-aadb428effd9

22. Soham, B. (22.8.2024). What is React Speech Recognition and How to Implement it in Your Apps. Haettu 3.12.2024 osoitteesta https://reverieinc.com/blog/implementing-react-speech-recognition-in-your-apps/

23. Supraja. (17.2.2024). Functional Components Vs. Class Components in React. Haettu 2.09.2024 osoitteesta https://tinyurl.com/mryf9b88

24. TailwindCSS. (n.d.). Get started with Tailwind CSS. Haettu 15.10.2024 osoitteesta https://tailwindcss.com/docs/installation/using-vite

25. TailwindCSS. (n.d.). Responsive Design. Haettu 15.10.2024 osoitteesta https://tailwindcss.com/docs/responsive-design